REMARKS/ARGUMENTS

By way of an Office Action dated February 25, 2005, the Examiner objects to the Specification. In response, Applicant has amended the Abstract to remove the improper content (File location) in line 11.

Further, Applicant has properly numbered the Provisional Application from which this patent claims priority as suggested by the Examiner.

Applicant has amended claims 17 and 33 so as to correct simple typographical errors. These amendments are **not** narrowing amendments made for the purposes of patentability. Rather, the amendments simply correct the spelling of the words dictorial and dcitorial to dictatorial in both claims.

Claims 1 through 38 were rejected under 35 U.S.C. § 102.

Concerning this rejection, the Office Action solely relies upon U.S. Letters Patent 5,359,730 (hereinafter "Marron") to anticipate Claims 1 through 34. Applicant believes that as an initial matter, the Examiner should be aware of the significant distinctions between Marron and the present invention. Marron is directed to "replacing old operation system programs or modules with new updated versions thereof while providing continuous availability and operations of the system." (Abstract). This is performed on the system where there exists complete, separately executing, software applications. Marron does not modify the old program, but rather routes processes from the old program to a separate and independently executing

new program. (Col. 7, lines 29-31). Marron clearly distinguishes between the old program, new program, and processes routed from the old program to the new program. Marron does not suspend the old program and does not suspend the new program. At best, Marron places the process in a wait state to determine whether to route the process from the old to the new separate program. (Col. 7, lines 39-40).

In contrast to Marron, the present invention could be used to suspend the "old program" of Marron, update the "old program", and resume the "old program." By using the present invention, the functionality of Marron is not needed since there is not a "new program" in the present invention, and the present invention does not route processes from one independent and separately executing program to another.

Marron does not anticipate Claims 1 through 34 because Marron does not claim or disclose the creation of initial source code nor segmenting the initial source code. Therefore, Marron cannot anticipate Claims 1 or 19, nor the respective dependent claims. The present invention, however, is used to create an initial version of source code, segmenting said initial version of source code into grains and translating the initial source code to object code. Marron only discloses the creation of a new program which is not the initial version of source code. "Initial" means first and Marron does not claim or disclose the creation of the first version, i.e., the "old program" of Marron.

The Office Action states that Marron anticipates the independent claims element of segmenting said "initial grains" by Marron's reference to "change modules."

However, Marron does not claim or disclose any segments or segmenting of any software application. Further, Marron does not claim or disclose segmenting software into grains as in the pending application. Marron, conversely, contemplates the replacement of entire modules and programs as Marron specifically states: "The general problem that the invention addresses and solves is how to **replace** modules ...." (Marron, col. 6, lines 26-28). The invention of Marron installs "new modules and programs." (Marron, col. 6, line 55). The invention of Marron determines when to "stop executing the old program and start executing the new program." (Marron, col. 7, lines 5-7). Marron also "loads new copies of the new programs A' and B'." (Marron, col. 7, lines 55-56). Specifically, Marron requires that the programmer who wishes to implement changes to a mainframe operating system must create new programs, recompile the new programs, and link the new programs to form new programs. (Marron, col. 6, lines 50-54). Marron does not modify the old program, but rather states that the process (not the old or new program) will be "routed either to the old code if the process is unsafe or to the new code if the process is safe." (Col. 7, lines 30-31). Marron's determination of when a process is to be routed from the old program to the new program does not anticipate modifying the old or new program to the new program. Marron has the old program fully executable and the new program executable and processes are routed between the old program to the new program.

Marron does not suspend the old program in mid execution, but rather routes processes.

Marron does not anticipate mapping first version grains onto second version grains.

The Office Action states that Marron anticipates mapping said first grains onto second version grains through the language of "the new versions are loaded into the system along with change instructions providing information controlling the update." (Abstract). These instructions do not modify the old program, but route processes from the old to the new program.

Since Marron does not disclose the segmenting of software into grains as stated above, it is impossible for Marron to anticipate mapping first version grains onto second version grains. Marron is directed to having new programs created by a change programmer modifying the old programs recompiling and linking the new programs to from load modules. These load modules contain change instructions for dynamically installing the new modules and programs. (Marron, col. 6, lines 50-55). The function of mapping any lines or segments of code from one version to the other is not claimed or disclosed in the Marron patent.

Further, because Marron routes applications from the complete and executable old program to the complete and executable new program, there is no incentive nor need for Marron to map from one program to another. (Col. 7, lines 28-34). Thus, the present application, which is directed to creating software that is modifiable without halting its execution via the use of segments known as grains, is not anticipated by Marron.

Further, Claims 11 and 29 require mapping said first grains onto said second grains. Claims 11 and 29 require the creation of a dynamic list of first grains and corresponding second grains for at least those first grains to be modified according to said second version of source code. Again as stated above, Marron does not teach the use of grains as discussed above. As Marron's teaching of new modules or change modules does not teach the segmentation of software into grains, Marron cannot claim or disclose mapping first version grains with second version grains.

The Office Action relies upon Marron's "change-instructions" and "change descriptor" language to reject Claims 2, 3, 7, 8 and corresponding subsequent claims. However, this language of Marron is very different than the present invention. Marron uses this language to describe when a task is "safe" to be routed to the new program. (Col. 7, lines 61-66). When the task is safe, it is routed to the new program. (Col. 7, lines 28-31). The old program is not suspended. Marron discloses "change-instructions" and "change descriptor" that provide functionality for routing the task. Marron, unlike the present invention, does not claim or disclose the display of grain boundaries to a computer programmer. Therefore, Claims 2, 3, 7, 8 and the corresponding subsequent claims are not anticipated by Marron.

Marron does not anticipate a hot pack.

The Office Action states that Marron anticipate the hot pack through its language "new programs are created". However, Marron state that the new program is created for replacement of the old program existing on the mainframe operating

system. Specifically, Marron states that "the new programs are created by change programmer modifying the old programs, recompiling, and linking the new programs to form load modules." (Marron, Col. 6, lines 50-54). The load modules contain change instructions for "dynamically installing the new modules and programs." (Marron, Col. 6, line 55). The invention of the present application, however, directs modification to the target software application (at best, the old program) through replacement grains (not disclosed in Marron) targeted to the target software application contained in the hot pack (not disclosed in Marron), rather than routing tasks from old programs to new programs. Therefore, Marron does not anticipate a hot pack since Marron is limited to replacement of new programs with old programs and does not claim or disclose modification of first version grains of a target software application.

Marron's use of the terms "old program" and "new program" does not equate to first or second version grains.

The Office Action states that Marron anticipates "first version grains" through the disclosure of "modifying old programs." However, the old program of Marron is only modified separate from execution and the new program created without regard to execution of the old program. The present invention modifies the target application during the target application's execution and does not route process from an old program to a new.

"First version grains" are segments of the target software application (old program, at best, of Marron). Specifically, the pending application contains the

following: "Grains delineate the source code and object code into discreet segments...." Further, "modifying the old programs", as stated in Marron, refers to the creation and replacement of the separate old program with a separate new program. (Marron, Col. 6, lines 55-56). These new programs of Marron are complete programs so that the old programs are replaced. (Marron, Col. 6, lines 21-26). The present invention does not have two separate programs, old and new. Rather, the present invention, as claimed in the independent claims, modifies the target software application (at best, the "old program" of Marron) through the use of a hot pack. The present invention does not route a task from an "old program" to an entirely separate "new program."

Marron's purpose is "replacing old operating system programs or modules with new updated version...." (Marron, Abstract). Marron does not modify the executing "old program", but replaces it. Further, Marron must determine when to execute the old programs or when to execute the new programs. (Marron, Abstract and Fig 2B, step 64 "Switch Over To New Programs" and Fig 3, steps 72, "Route to New Program" and step 74, "Route to Old Program"). Marron does not modify an executing application.

Further, the Office Action states that Marron anticipates "second version grains" through the disclosure of "new programs." A "second version grain" of the present invention is a segment of the target software application, second version grains are not separate and independent software applications, but are replacement segments of

the target software application, not the "new program" of Marron. Therefore, Marron does not claim or disclose a second version grain. The modification of a first grain according to a second grain is not claimed or disclosed in Marron.

Marron does not claim or disclose the modification of a first version grain to a second version grain according to a dictum. The Office Action states that "determining the status of said dictum" is anticipated by Marron's language, "to determine ... whether program A or program A' should be executed." However, the determination as to whether to execute program A or A' of Marron contemplates routing of a process from A to A" (Col. 7, lines 30-31). This determination does not claim or disclose the modification of the program A while it is executing. This determination does not claim or disclose modification of the target software application of the present invention. In Marron, the new program A' needs to have been created, complied, installed and existing on the mainframe multiprocessor operating system prior to the determination as to whether to route a process from program A to program A'. The present invention, however, can modify the target software application while it is executing and while a process is already running the target software application.

The dictum of the present invention determines when and how the modification of the first version grain should occur. The dictum of the present invention does not determine which of two simultaneously existing programs (A and A') should have a process routed to it. Therefore, Marron does not anticipate the creation of a dictatorial

to determine whether to modify the first version grain according to a second version grain as stated in independent Claims 11 and 29.

Marron's "safety points" do not anticipate crumbs.

Marron's description of "safety points" does not claim or disclose crumbs of the present application. Safety points are defined in Marron as "conditions [that] can be translated into events in the life of a task, or the combination of an event with an observable state of the process." (Marron, Col. 7, lines 7-9). Respectfully, the safety points of Marron that may be in the old program (Col. 7, lines 34-36) do not show that the old program will be suspended. Specifically, the reference to the "wait" in Marron states that "a 'wait' instruction that places the process in a wait state." (Col. 7, lines 39-40). A process, according to Marron, is a task being performed by the system. The process will be "routed either to the old code if the process is unsafe or to the new code if the process is safe." (Col. 7, lines 30-31). The process which is placed in a "wait" state is all that is disclosed by Marron. (Col. 7, lines 39-40).

Specifically, Marron states that "multiple tasks and processes can independently access the programs." (Col. 1, lines 19-20). Therefore, any mention by Marron of placing processes in wait states does not equate to suspending the old or new program and, therefore, does not anticipate suspending the target software application. Respectfully, Marron's "wait" does not place the target software application in a suspended state, but only places a process in a wait state.

Since Marron expressly states that "a 'wait' instruction that places the process in a wait state" (Co. 7, lines 39-40) and the process is routed to the old or new program (Col. 7, lines 30-31), the process is separate and apart from the old or new program. Therefore, Marron's disclosure of placing the process in a wait state cannot be equated to placing the old or new program in a wait state. Marron specifically requires the old and new program to continuously execute since the mainframe of Marron has multiple tasks and processes that can be routed to the old or new program. Marron does not suspend the old or new program.

Placing the old or new program in a wait state is non-sensical given the purpose of Marron. Marron is directed to "replacing old operation system programs or modules with new updated versions thereof while providing continuous availability and operation of the system." (Abstract). To suspend the old or new program would not provide continuous availability and operation of the system. However, placing the process in a "wait" state prior to selecting whether to route to the old or new program would allow continuous availability and operation of the old and new applications of Marron.

Respectfully, the "wait instruction" places the process in a wait state, not the entire old and new programs. Therefore, Marron does not anticipate suspending the target software application. Further, Marron itself notes that "safety points most often reside in modules which are not being changed." (Col. 2, lines 29-31). This is consistent with Marron's use of safety points to place processes in wait states, but is

not consistent with placing the old or new program in a wait state. The present invention suspends the target software application for modification. Marron places the process in a wait state and does not suspend the old program, but routes the process to the new program.

Marron does not claim or disclose crumbs.

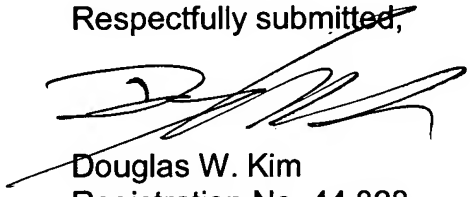
Crumbs are physical locations within a grain that can be used to determine when the execution pointer hits a predetermined point in the execution of the target software application. Marron specifically describes safety points as "when a task is: started after the change was implemented, ...entering or exiting a particular module, ...making a particular system call, executing an instruction at a given offset at a given module...observed as being in a problem or user state...observed as being in a wait state...running under a job name, and not running under a given job name." (Marron, col. 7, lines 12-24). Further, safety points are "either observed by the system since they include a system call, or observed by the DSUF...." (Marron, col. 7, lines 25-27). Marron's "safety points" do not anticipate crumbs because crumbs are locations in the target software application that can allow dictums to be evaluated once reached by the execution pointer. Therefore, the "safety points" of Marron does not anticipate crumbs.

Although the arguments above are principally directed to the independent claims, they are equally applicable to the remaining dependent claims. For the reasons presented herein, Applicant respectfully requests that this application be

granted a notice of allowance for Claims 1 through 38 in the normal course of Patent Office business.

In the event that the Examiner does not find these amendments and arguments persuasive, the Applicant requests an interview to discuss the Office Action and claims of this pending application.

Respectfully submitted,



Douglas W. Kim
Registration No. 44,828
McNAIR LAW FIRM, P.A.
P.O. Box 10827
Greenville, SC 29603-0827
Telephone: 864-232-4261
E-mail: dkim@mcnair.net
Attorneys for the Applicant